Discovering High-Quality Chess Puzzles with Offline Reinforcement Learning

Allen Nie[1,*], **Anirudhan Badrinath**[1,*], **Nicholas Tomlin**[2,*], **Timothy Dai**[1], **Carissa Yip**[1], **Rose E Wang**[1], **Emma Brunskill**[1], **Chris Piech**[1]
anie@cs.stanford.edu, {ebrun,piech}@cs.stanford.edu

[1]**Stanford University**
[2]**UC Berkeley**

[*] Equal contribution

Abstract

Learning and skill mastery require extensive and deliberate practice. In many learning settings, producing high-quality pedagogical materials can require a high level of domain expertise and be very time-consuming. Pedagogical materials often need to train students to engage in different thinking patterns. In some domains, such as chess, puzzles are used to help students practice their skills in calculating the next moves and recognizing known patterns on a board. Giving students a practice set of puzzles to help them learn different modes of thinking is challenging because the teacher needs to carefully balance between different motifs and how many look-ahead steps a student needs to perform. Popular online platforms like Chess.com and Lichess offer players millions of puzzles. Unlike chess tactics puzzles procured by human experts, where chess beginners can learn valuable insights, these puzzles are automatically generated and often regarded as having low pedagogical value. These platforms also rely on a heuristic to recommend puzzles to users for practice. Using the user history data over an entire year, a total of 1.5 billion puzzle-solving histories, we learn the pedagogical value of a puzzle and how to automatically choose a set of puzzles to better support chess learners using insights from offline reinforcement learning. We show that using offline policy evaluation, our trained policy has significant impact on beginners with puzzle-solving Elo range of 100–1000, particularly for the group of beginners whose learning growth was stagnant. We also performed a qualitative analysis of the puzzles discovered by our model by collecting annotation ratings from expert chess players. The success of our pipeline shows promise for a future where we can understand the pedagogical values of practice items given general user interaction data.

1 Introduction

Practice makes perfect. The foundation of acquiring knowledge or mastering a new skill relies on countless hours of mindful and deliberate practice (Ericsson et al., 1993; Ericsson, 2008). Koedinger et al. (2023) suggests that across many different learning settings, when students are provided with high-quality, curated learning materials, they can all learn at a similar rate and achieve success with extensive practice. However, the creation of high-quality learning materials is often a key bottleneck. Though lectures can be recorded in video and knowledge can be transcribed in text, students still need to be able to practice what they have learned through forced retrieval and synthesis, which has been shown to greatly enhance learning (Roediger III et al., 2011). Producing a large amount of practice materials often requires significant human expertise and heavy time investment. It is also difficult to evaluate the pedagogical value of each learning material in knowledge acquisition.

In chess, players often learn by playing against each other directly. However, they also learn important skills through tactics books, which are comprised of puzzles – subgames limited to a few moves to teach important concepts or to train players to plan multiple moves in a sequence in order to gain more advantage over their opponent. These puzzles are very common for beginners and are considered good learning materials because they isolate and highlight difficult concepts into a small subgame and prime beginners into a habit of thinking strategically (Henkin, 2021). Online chess platforms such as Chess.com and Lichess provide puzzles for players to practice. In order to keep their players engaged, these online platforms aim to serve fresh puzzles on a weekly basis so that players will always have new materials for learning and fun. Chess.com serves roughly 441K puzzles to their players and Lichess hosts over 3.8M puzzles.

In order to produce a large number of new puzzles quickly, unlike classic tactics books where puzzles are curated by human experts through careful deliberation, review, and editing, online chess platforms use an algorithm to automatically generate puzzles from the actual games played by players. The puzzles are assigned an Elo rating and presented to players at random, provided their difficulty is within a fixed range of the player's rating (Chess.com, 2023). However, it remains unclear whether these puzzles effectively contribute to a player's long-term chess learning. In a system where learning materials are automatically generated, it is essential to filter out content that fails to produce meaningful, lasting gains in knowledge or skill.

Offline reinforcement learning (RL) can learn a policy from a dataset of historical interactions. It has been used to discover patient treatment policies in the ICU (Komorowski et al., 2018; Luo et al., 2024), to create personalized learning paths in math education software (Mandel et al., 2014; Bassen et al., 2020; Ruan et al., 2024), and to learn new controllers for robotics (Kumar et al., 2020; 2021; Lu et al., 2022). It has been shown to scale with data and can learn policies that generalize beyond the training distribution (Kumar et al., 2022).

Using a large dataset of 1.5 billion puzzle solving attempts over the course of a year from Chess.com with 3.1 million users, we can use an offline RL algorithm to learn a policy and evaluate its effectiveness using a holdout portion of the dataset. We first show that, similar to the finding in Wang et al. (2022), we can separate users into two groups: a growth group, where the user's Elo rating gradually increases with more puzzles, and a stagnant group, where the user's Elo remains flat. We then propose a simple advantage-weighted actor-critic objective for offline policy learning based on Nair et al. (2020); Kostrikov et al. (2021) and use continuous action embeddings to account for the large action space. Finally, we show that, estimated with one-step importance sampling, the policy learned to serve puzzles significantly more effectively for beginners (Elo[1] 100–1000) than the original Chess.com system. Our qualitative evaluation provides preliminary evidence that the learned policy recommends puzzles that may be slightly more fun and somewhat harder relative to the user's rating.

2 Related Work

There is a long history of using games such as chess to evaluate the progress of AI (Campbell et al., 2002; Silver et al., 2017; 2018). Although these models have developed superhuman abilities to master the game, few investigations have focused on leveraging their knowledge to teach humans. Schut et al. (2023) and McGrath et al. (2022) did pioneering work on uncovering chess knowledge learned in AlphaGo and AlphaZero. McIlroy-Young et al. (2021) built models to identify the chess playing styles of each user and then later released models that imitate chess players of different levels (McIlroy-Young et al., 2022). Hamade et al. (2024) built a chess agent that can match a weaker player's skill to foster skill-compatible learning. However, no end-to-end system has been developed to recommend chess puzzles specifically for teaching humans.

In automated teaching systems, reinforcement learning has been used to adaptively select instructional materials for students. Chi et al. (2009) built an adaptive tutoring system for simple decision

[1]Throughout this paper, we use "rating", "Elo" to refer to the puzzle Elo score established by Chess.com: https://www.chess.com/leaderboard/tactics. Both user and puzzles are assigned an Elo score. Both change initially, but after a while, puzzle's Elo score becomes fixed and ceases to change. We do not use any information outside of tactics.

making in college math education. Ruan et al. (2024) built a math tutoring chatbot that decides when and how to provide hints. Mandel et al. (2014) uses offline RL to augment the fraction learning experience of students in math games. Nie et al. (2023) examines how RL-based personalization in math tutoring systems differentially impacts subgroups of students. Liu et al. (2022) leverages meta-exploration to give feedback on interactive student programs. However, most of these projects remain small in scale, and offline RL has not demonstrated usefulness for large-scale educational settings. In our work, we follow Kumar et al. (2022) to scale offline RL models to large datasets with billions of interactions and use offline RL to discover high-quality puzzles that can improve a chess player's learning experience.

3 Data

Our dataset consists of players' puzzle history data from 3,132,428 unique, active users of a popular chess website, Chess.com, playing a total of 1,536,254,297 puzzles (441,113 unique puzzles) over the course of one year from March 2021 to March 2022. On average, a user played 490.4 puzzles over this one year. Of those 490.4 puzzles, 96.9% are played within three minutes of another puzzle, which indicates that players tend to play puzzles in short bursts of time. An average burst comprises 5.1 games. Playing in bursts naturally suggests that users do not continuously play puzzles throughout the day. In fact, the average time between these bursts is approximately two and a half days. In general, our dataset reflects the engaged and extensive user base of online chess players.

Puzzle Serving According to Chess.com administrators, the site serves puzzles to users using a bucketed uniform policy. Both puzzles and users are assigned an Elo score, specifically Chess.com's tactics/puzzle-solving rating rather than a rating from games against other humans (see footnote in Section 1). This policy serves a player's next puzzle by first bucketing all chess puzzles that have puzzle Elo ratings within ± 200 points of a player's Elo and then sampling uniformly from that bucket. As a player begins to fail puzzles, that ± 200 bucket eases to $-300/+100$ after one incorrect puzzle, $-400/+0$ after two, $-500/-100$ after three, and $-600/-200$ after four incorrect puzzles. In this way, the Chess.com policy adapts to player performance by serving progressively easier puzzles in response to a player's difficulty in solving previously served puzzles. Overall, Chess.com's *bucketed uniform policy* integrates a player's performance on their past four puzzles, their Elo rating, and puzzle rating to determine the next puzzle served.

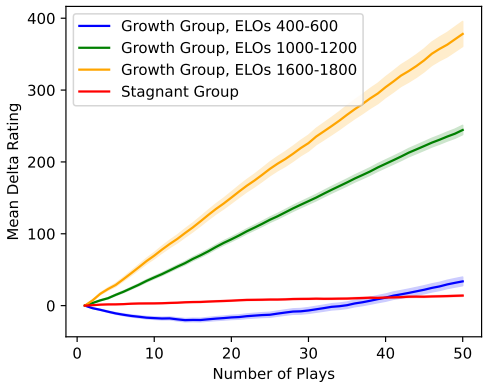
Chess Puzzles Chess puzzles are core units of learning. In a chess puzzle, the player is presented with an initial position and tasked with finding the correct solution in one or more moves, adhering to the puzzle's main goal or task, described in part through a puzzle's motifs. The initial position is shown to the player as a board, and it can also be conveniently encoded as a FEN (Forsyth-Edwards Notation) string. Move counts for puzzles in our dataset range from 1 to 16, averaging at 2.6. Harder puzzles tend to demand more moves, with the puzzle's difficulty reflected in its puzzle rating. Our puzzles' ratings range from 100 to 4000. In contrast to move count and puzzle rating, which are both numerical, motifs provide more qualitative insights into the diversity of puzzles.

A puzzle's motifs describe the primary strategy or skill required to solve the puzzle. Examples of such motifs include "Exchange Sacrifice" and "Promotion," describing a notable event that takes place in a puzzle. A puzzle can have more than one motif; in fact, the puzzles in our dataset have, on average, 5.6 motifs each, with motif count ranging from no motifs at all to 56 motifs. Of our 441,113 puzzles, 26% do not have any motifs at all. Thus, motifs provide informative yet nonexhaustive insights into the quality and variety of served puzzles.

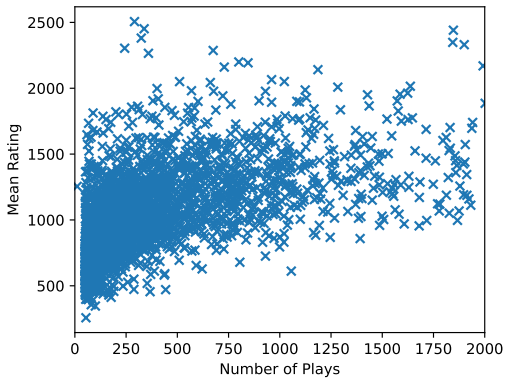
Analyzing User Elo Growth In Figure 1a, we plot the mean change in rating across the first 50 recorded puzzle plays for four groups of players based on their Elo ranges and relative increases in Elos. The "Growth Group" represents players in the 99th percentile for Elo increase (the change in Elo from each player's first recorded Elo) among players in their Elo range, while the "Stagnant Group" represents players in the 1st percentile for Elo increase, combined across all three Elo ranges. The trajectory of the Stagnant Group, as expected, is relatively flat, signifying little growth in Elo across the first 50 games played. On the other hand, players in the Growth Group of high Elos

improve rapidly in their first 50 games, with less rapid growth for lower Elo players, and even less rapid growth for the lowest Elo players. This suggests that a player's average, long-term Elo generally points to their initial improvement speed. Interestingly, the Growth Group of the lowest Elo players experiences a dip in Elo in their first ∼40 games, falling below Elo improvements of even the Stagnant Group, before seeing gains. This dipping trend suggests that, for overall inexperienced players, playing chess puzzles involves an especially difficult acclimatization period, during which unfamiliarity with the game may initially hinder performance, but players eventually learn the rules and complexities of the game and reap the benefits of playing more puzzles.

We broaden our analysis in Figure 1b and examine players' total puzzle counts. Each point in the figure represents an individual player's total number of puzzles played. We observe a general trend from the bottom left toward the upper right, indicating that a higher number of puzzles played generally correlates with a higher Elo. This strengthens the suggestion that playing chess puzzles is a skill that improves with more experience. The total number of puzzles played is generally an indicator of a player's skill level.



(a) Plot illustrating the mean change in rating for four groups of players based on their Elo ranges and their relative increases in Elos.



(b) Scatter plot showing the relationship between a player's puzzle play count and their rating averaged over a year.

Figure 1: Plots that illustrate user Elo growth, showcasing the Elo growth trajectories of several groups of players (Figure 1a) alongside overall Elo trends across our entire player base (Figure 1b).

4 Policy Learning

4.1 Preliminaries

We define a stochastic decision process $M = \langle \mathcal{S}, A, \mathcal{T}, r, \gamma \rangle$, where $\mathcal{S}$ is a set of states; A is a set of discrete actions; $\mathcal{T}$ is the transition dynamics; r is the reward function; and $\gamma \in (0, 1)$ is the discount factor. Let $\mathcal{D} = \{\tau_i\}_{i=1}^{n}$, where $\tau = \{(s_j, a_j, s'_j, r_j)\}_{j=0}^{H-1}$, with $s'_j \equiv s_{j+1}$, denote a dataset of trajectories collected under some policy on M with time horizon H. We denote the performance of a policy π based on its expected discounted return $R_t = \mathbb{E}_{\tau\sim\rho_\pi}[\sum_{t'=t}^{H-1} \gamma^{t'-t} r_{t'}]$ where ρ_π is the distribution of τ under policy π. By the definition of the action-value and value function respectively, we let $Q(s, a) = \mathbb{E}_{\tau\sim\rho_\pi}[R_t \mid s, a]$ and $V(s) = \mathbb{E}_{\tau\sim\rho_\pi}[R_t \mid s]$.

In an off-policy policy learning problem, we do not have access to the true transition dynamics $\mathcal{T}$ or any online interaction with M. Instead, we take an offline dataset $\mathcal{D}$, which can be collected by one or a group of distinct policies, which we collectively refer to as the behavior policy π_b on the decision process M.

In actor-critic frameworks, we perform policy evaluation and policy improvement in conjunction to derive an optimal policy π^* (Konda & Tsitsiklis, 1999). Typically, policy evaluation is performed using iterative application of the Bellman update. In the context of deep RL for off-policy learning (i.e., where we have $\mathcal{D}$, policy π_θ parameterized by θ, action-value function parameterized by

ϕ), we perform policy improvement through gradient updates to the policy π_θ and optimize Equation 1.

$$\arg\max_\theta \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}[Q^\pi_\phi(\mathbf{s},\mathbf{a})] \tag{1}$$

For our formulation, we consider a non-Markovian policy π_θ based on a transformer architecture, where the state and action space incorporate a fixed length context. Specifically, our state space consists of a continuous embedding-based k-dimensional representation of the user's puzzle history and learning progress (e.g., Elo), i.e., $s \in \mathcal{S} = \mathbb{R}^k$. Our desired action space is the set of all $N = 441{,}113$ chess puzzles, i.e., $a \in \mathcal{A} = \{1, 2, \ldots, N\}$.

In an off-policy setting, common pitfalls with traditional actor-critic techniques include extrapolation error by venturing outside of the supported data distribution in $\mathcal{D}$, which results in an accumulation of errors from bootstrapped action-value functions (Sutton & Barto, 1998). To ensure that the trained policy π_θ stays close to the behavior policy, we penalize the statewise Kullback-Leibler divergence $D_{\mathrm{KL}}(\pi_\theta(\cdot \mid \mathbf{s}) \,\|\, \pi_b(\cdot \mid \mathbf{s}))$, averaged over states sampled from $\mathcal{D}$, with coefficient β (Rafailov et al., 2024; Jaques et al., 2017).

$$\arg\max_\theta \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}\left[Q^\pi_\phi(\mathbf{s},\mathbf{a})\right] - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot \mid \mathbf{s}) \,\|\, \pi_b(\cdot \mid \mathbf{s}))\right] \tag{2}$$

4.2 Deriving an Offline Policy Learning Objective

We derive a simple advantage-weighted actor-critic-style objective for offline policy learning, based primarily on Nair et al. (2020) and Kostrikov et al. (2021). We leverage offline advantage estimation via a parameterized value function $V^\pi_\psi(\mathbf{s})$, augmenting the objective shown in Equation 2 with a baseline. Importantly, note that since the value function and its inputs are constants with respect to the optimization variable, θ, it does not bias or modify the objective.

$$\arg\max_\theta \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}\sim\pi_\theta(\cdot|\mathbf{s})}\left[Q^\pi_\phi(\mathbf{s},\mathbf{a}) - V^\pi_\psi(\mathbf{s})\right] - \beta D_{\mathrm{KL}}(\pi_\theta(\cdot \mid \mathbf{s}) \,\|\, \pi_b(\cdot \mid \mathbf{s}))\right] \tag{3}$$

Based on the derivation in Nair et al. (2020), we can project the closed-form optimal solution $\pi^*(\mathbf{a}|\mathbf{s}) \propto \pi_b(\mathbf{a}|\mathbf{s}) \exp(\frac{1}{\beta}(Q^\pi_\phi(\mathbf{s},\mathbf{a}) - V^\pi_\psi(\mathbf{s})))$ into the policy space by minimizing $D_{\mathrm{KL}}(\pi^*(\cdot \mid \mathbf{s}) \,\|\, \pi_\theta(\cdot \mid \mathbf{s}))$ over the state distribution, which yields the weighted maximum likelihood objective shown in Equation 4.

$$L_\pi(\theta) = \mathbb{E}_\mathcal{D}[-\log \pi_\theta(\mathbf{a}|\mathbf{s}) \exp(\frac{1}{\beta}(Q^\pi_\phi(\mathbf{s},\mathbf{a}) - V^\pi_\psi(\mathbf{s})))] \tag{4}$$

Mirroring Kostrikov et al. (2021), we train the value network and action-value network for advantage estimation purely using transitions in the dataset $\mathcal{D}$. To train the value network, we leverage expectile regression, as shown in Equation 5, and to train the action-value network, we use the objective shown in Equation 6.

$$L_V(\psi) = \mathbb{E}_\mathcal{D}[L_2^\tau(Q^\pi_\phi(\mathbf{s},\mathbf{a}) - V^\pi_\psi(\mathbf{s}))] \tag{5}$$

$$L_Q(\phi) = \mathbb{E}_\mathcal{D}[(r(\mathbf{s},\mathbf{a}) + \gamma V^\pi_\psi(\mathbf{s}') - Q^\pi_\phi(\mathbf{s},\mathbf{a}))^2] \tag{6}$$

4.3 Architecture for Chess Puzzle Recommendation

Since we formulate the recommendation problem as an offline policy learning problem, we use a causal transformer architecture as our decision making policy, denoted by T_{θ_T}. We choose this architecture because it is widely used for student knowledge tracing and modeling in education (Pandey & Karypis, 2019; Choi et al., 2020; Shin et al., 2020). At each sequence element (i.e., time step t), the transformer takes as input the concatenation of a user representation u_t and a puzzle representation p_t, each constructed by applying a neural network to the corresponding user and puzzle features. The outputs at each sequence element are (a) a prediction of the next puzzle, $\pi(\mathbf{a} \mid \mathbf{s})$, and (b) a value function estimate, $V(s)$.

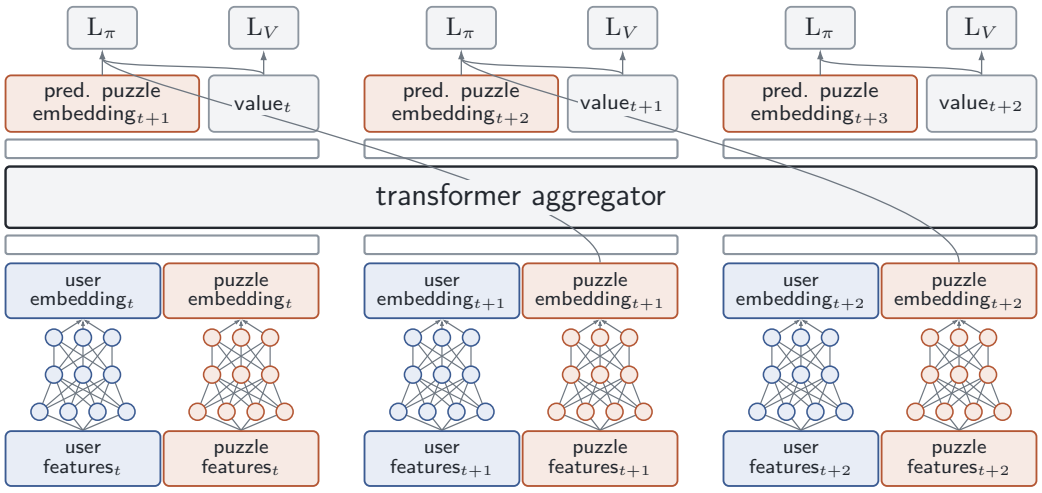


Figure 2: Architecture and training procedure of the chess puzzle recommendation policy, leveraging user features and puzzle features to optimize a policy-based loss L_π and a value function via L_V.

During training, we optimize L_π, L_Q, L_V on predictions at each sequence element, with training sequences sampled uniformly from the chess puzzle interaction data. Importantly, we apply teacher forcing during the forward pass of training, which is standard in training autoregressive decoder-only transformers.

User Embedder To construct a latent representation of the user, we embed user-specific features f_u, i.e., the user Elo and user correctness at time t, using a small, two layer multi-layer perceptron (MLP), denoted as U_{θ_U}. We normalize the output user embedding $u_t = U_{\theta_U}(f_u)$ to unit norm (e.g., L2 normalization).

Puzzle Embedder To construct a latent representation of the puzzle, we embed puzzle-specific features f_p using an MLP and convolutional neural network (CNN), denoted as P_{θ_P}. Specifically, we use a CNN to encode the board position associated with the puzzle, alongside a learned embedding for each individual puzzle and its first move. Similarly to the user embeddings, we normalize the puzzle embedding $p_t = P_{\theta_P}(f_p)$ to unit norm (e.g., L2 normalization).

Practical Considerations for Policy Learning Given a large discrete action space $\mathcal{A}$ (i.e., approximately half a million puzzles), computing and normalizing the policy probabilities $\pi_\theta(a \mid s)$ across the entire action space is intractable during training. We therefore parameterize the policy using an exponentiated inner product between the transformer's output and each candidate puzzle embedding. Let $h_t = T_{\theta_T}(\mathbf{p}_{t-T:t}, \mathbf{u}_{t-T:t})$ denote the transformer's output for the history through time t, and let $p_a = P_{\theta_P}(f_a)$ denote the embedding of candidate puzzle a. Equation 7 defines the resulting temperature-weighted softmax policy.

$$\pi_\theta(a \mid s_t) = \frac{\exp\left(\lambda h_t^\top p_a\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\lambda h_t^\top p_{a'}\right)} \tag{7}$$

Although puzzle embeddings can be precomputed, evaluating the denominator in Equation 7 over the entire action space remains expensive during training. We therefore approximate it using the puzzle embeddings $P(B)$ in minibatch B as candidate negatives. For the observed next puzzle a_{t+1} with embedding p_{t+1}, the approximation is

$$\hat{\pi}_\theta(a_{t+1} \mid s_t) = \frac{\exp\left(\lambda h_t^\top p_{t+1}\right)}{\sum_{p_n \in P(B)} \exp\left(\lambda h_t^\top p_n\right)} \tag{8}$$

Afterwards, we apply the losses L_π, L_Q and L_V to optimize the policy with respect to the transformer parameters and user and puzzle embedders. We summarize the entirety of the training algorithm in Algorithm 1 and as depicted in Figure 2.

Algorithm 1 Offline training algorithm for chess puzzle recommendation

Input: $\mathcal{D}$, T_{θ_T}, U_{θ_U}, P_{θ_P}
for each minibatch $B \subset \mathcal{D}$ **do**
$f_p, f_u \leftarrow B, p_t \leftarrow P_{\theta_P}(f_p), u_t \leftarrow U_{\theta_U}(f_u)$ ▷ puzzle and user features
$\hat{\pi}_\theta(\mathbf{a}_{t+1} \mid \mathbf{s}_t) \leftarrow \frac{\exp\left(\lambda\, T_{\theta_T}(\mathbf{p}_{t-T:t}, \mathbf{u}_{t-T:t})^\top p_{t+1}\right)}{\sum_{p_n \in P(B)} \exp\left(\lambda\, T_{\theta_T}(\mathbf{p}_{t-T:t}, \mathbf{u}_{t-T:t})^\top p_n\right)}$ ▷ in-batch approximation
$L_V(\psi) \leftarrow \mathbb{E}_{\mathcal{D}}\left[L_2^\tau\left(Q_\phi^\pi(\mathbf{s}, \mathbf{a}) - V_\psi^\pi(\mathbf{s})\right)\right]$
$L_Q(\phi) \leftarrow \mathbb{E}_{\mathcal{D}}\left[\left(r(\mathbf{s}, \mathbf{a}) + \gamma V_\psi^\pi(\mathbf{s}') - Q_\phi^\pi(\mathbf{s}, \mathbf{a})\right)^2\right]$
$L_\pi(\theta) \leftarrow \mathbb{E}_{\mathcal{D}}\left[-\log \hat{\pi}_\theta(\mathbf{a} \mid \mathbf{s}) \exp\left(\frac{1}{\beta}\left(Q_\phi^\pi(\mathbf{s}, \mathbf{a}) - V_\psi^\pi(\mathbf{s})\right)\right)\right]$
$(\theta_T, \theta_U, \theta_P, \phi, \psi) \leftarrow (\theta_T, \theta_U, \theta_P, \phi, \psi) - \eta\nabla(L_V + L_Q + L_\pi)$
end for

4.4 Deriving a Reward Function

During evaluation, we compute puzzle embeddings once for the entire action space in order to obtain the policy's probability distribution over puzzles. This is an upfront cost rather than a per-query one: the same puzzle embeddings are reused across all user sequences and do not need to be recomputed for each new prediction.

We construct a scalar reward function that attempts to quantify the learning benefit in terms of the user's correctness on the puzzle and relative puzzle difficulty. Trivially, a correct response on a challenging puzzle is ideal, and it highlights progression in terms of learning, warranting a large reward. In any cases where the user answers incorrectly, it demonstrates no verifiable evidence of learning, so we assign no reward in these cases. Otherwise, in cases of partial correctness, we assign rewards based on the proportion of correct moves and relative difficulty of the puzzle.

Given the number of total moves N, number of correct moves (from the user) c, and the Elo of the puzzle and user, we assign the reward based on the difficulty-weighted correctness, with the difficulty weight α being a hyperparameter.

$$r(s, a) = \frac{c}{N} \exp(\alpha \cdot (\text{Elo}_{\text{puzzle}} - \text{Elo}_{\text{user}})) \quad (9)$$

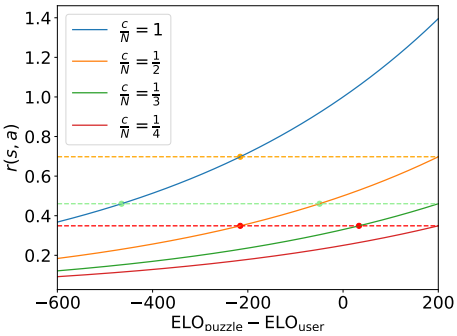


Figure 3: Reward function $r(s, a)$ for varying levels of correctness and Elo differentials between puzzle and user, with reward equivalences shown between balances of correctness and relative difficulty.

We provide a visualization of the reward function across different proportions of correct moves c/N in Figure 3 for $\alpha = 0.002$, which is the chosen hyperparameter due to its reasonable balance between partial correctness and difficulty. Specifically, based on internal testing, $\alpha = 0.002$ provides a reasonable level of reward equivalence between partial correctness levels (e.g., halfway correct) on challenging puzzles and complete correctness on easier puzzles (e.g., puzzles rated 200–400 Elo points lower). These equivalencies are displayed via dotted lines, where the color corresponds to the maximal reward achieved at a particular correctness level, and the intersection with other lines indicates the Elo delta at the reward equivalence.

5 Evaluation

While it would be ideal to run a randomized controlled trial experiment with the learned policy, there are many practical challenges. It is hard to recruit enough players to participate in the experiment, and it may take a significant time to be able to measure any difference reliably. We instead use a mixture of offline policy evaluation estimators and human evaluation to understand whether the learned policy could have meaningful impact on learning.

5.1 Evaluation Using Importance Sampling

To evaluate our trained policy with respect to the behavior policy, we can leverage importance sampling, using the known probability distributions of the behavior policy (e.g., as deployed by Chess.com) and our trained policy. Importantly, traditional importance sampling with sequences of size 256 (or even a fraction of that) is impractical due to the significant induced variance in the estimator over large sequences. Instead, we employ a one-step approximation (e.g., removing past weights) across 32 steps, which significantly reduces variance at the cost of being biased (Chen et al., 2019). We refer to this as one-step importance sampling. To further reduce variance, we clip the importance weight for each step $\pi_\theta(\mathbf{a}|\mathbf{s})/\pi_b(\mathbf{a}|\mathbf{s})$ between $\epsilon_s = 0.1$ and $\frac{1}{\epsilon_s} = 10.0$.

To quantify our improvements over the behavior policy for different users, we break down our results into different Elo groups. Additionally, we examine groups of stagnant growth and different growth groups, similarly to Figure 1; for this analysis, we use a simple definition of stagnant growth, which is growth of no more than 50 Elo points across the sequence. Importantly, this allows for examination of our improvements on different types of users and learning patterns, e.g., improvements on stagnant or new users may be more critical to retention on the Chess.com platform.

5.2 Expert Evaluation

Annotation Rubrics To annotate the quality of chess puzzles, we designed a detailed rubric with two chess experts involved in the project (USCF rating around 1900 and 2400). The rubric covers different characteristics of a puzzle, from whether it tests the player's ability to perform in-depth calculations of sequential moves or it focuses on testing the ability to recognize iconic patterns. In addition, the experts are also asked to judge if the puzzle is appropriate for a player of a given rating, whether it is a high-quality puzzle, and if it is fun to play. The rubric took multiple iterations with internal testing to verify its robustness and is described in full in Appendix A.1.

We recruited 8 chess experts to annotate 30 puzzles. The annotators were eight strong chess players, 5 of whom hold official chess titles. In total, there are 2 USCF Experts, 1 USCF National Master, 1 FIDE Master, 1 FIDE International Master, and 2 FIDE Grandmasters. The chess annotators' USCF Elo ratings are between 1990 and 2576. For a rough (not directly comparable) point of reference, the chess legend Magnus Carlsen has a FIDE rating of 2832, and the 100th highest-rated player has a FIDE rating of 2634 as of Spring 2025.

Category	Range
Calculation	1–4
Pattern	1–3
Informativeness	1–5
Appropriateness	1–5
Quality	1–5
Fun	1–5

Figure 4: Qualitative rating categories designed in collaboration with chess expert consultants.

The puzzles are randomly sampled from a *bucketed uniform policy* based on the player's rating range, and we have 6 players with different Elo scores. We chose 12 puzzles as *preference learning* puzzles that are annotated by all experts. This means that out of the 30 puzzles each expert annotated, only 18 puzzles are unique between them. The annotation was performed through an anonymized spreadsheet. Each annotator was asked to spend approximately 1.5 hours on the annotation and was later compensated with a gift card.

Scaling Annotations with LLMs Judging whether a chess puzzle is high quality or not requires annotators with significant expertise, who are distinctly different from standard crowd workers. This also means we are not able to perform massive annotations that can take up to hundreds of hours. However, sometimes, two policies that learn to recommend puzzles can behave very similarly and,

therefore, require a lot of annotations to understand if they are statistically significantly different. To help with scaling up the annotation effort, inspired by Zheng et al. (2023); He et al. (2023), we use the expert annotations to calibrate a large language model (LLM) that can imitate an expert chess annotator's judgments. LLMs seem to be able to understand FEN strings and can play chess with some deliberation (Toshniwal et al., 2022; Feng et al., 2023). Without making sweeping generalizations, we can leverage LLMs to provide preliminary understanding of whether the chess puzzles chosen by two systems are different.

With example expert annotations, we use DSPy (Khattab et al., 2023) to create our LLM chess puzzle judge. DSPy is an LLM library that allows us to specify a set of initial prompts, training data, and a metric. It then automatically selects the best expert examples to include in the LLM prompt that can maximize the metric. Internally, DSPy performs cross-validation with a random search to find the best subset of expert annotations. We learn 8 LLM annotators using the data from 8 chess annotators. We separately learn 8 annotators because each annotator's preference might be different, and imitating the behavior of 8 individuals seems more difficult than imitating the behavior of one individual. For each annotator, we use the 12 shared *preference learning* puzzle annotations described above, splitting them evenly into 6 training and 6 validation examples for DSPy prompt optimization.

Prompt Design We use a system message and the DSPy Signature module to design the prompt. The system message specifies the persona of the annotator using their USCF Elo rating and title. The DSPy signature contains three input fields—the annotation guidelines, the target user's Chess.com puzzle Elo, and the puzzle board—and one output field containing the six category scores. Appendix A.2 provides a complete description of these fields. For puzzle board representation, we experimented with a FEN string and a grid-like representation of the entire board. We found the grid-like representation to work better. Along with the board position, we also describe the first move of the puzzle in text.

6 Experimental Results

Training To train our chess puzzle recommendation policy, the dataset is partitioned randomly by users, with 90% used for training and 10% for evaluation. During training, each batch consists of 16 examples of users' puzzle histories, with a sampled sequence length of 256 to accommodate computational constraints. We use a discount rate of 0.99. During training, to implement Equation 8, we use 65K puzzles as the in-batch negative examples to estimate our policy's probabilities. Due to the large dataset size, we train the policy for 25K steps, alongside the learned action-value and value functions.

In this section, we examine the performance of our learned policy in two different ways. We use an offline policy evaluation method to provide an objective comparison between our policy's performance and the behavior policy's performance. Then, we use our automated annotation pipeline to provide another set of ratings.

6.1 Policy Evaluation

To demonstrate that our policy performs well qualitatively and quantitatively relative to the behavior policy, we perform evaluations based on importance sampling and visualize and summarize the policy recommendations. As a preliminary qualitative evaluation, we examine the distribution of puzzle Elos recommended by our policy with respect to the puzzle chosen by Chess.com's current policy in Figure 5. As a preliminary sanity check, our policy consistently recommends puzzles in the rough ballpark of the chosen puzzle and the user Elos, though our policy tends to recommend harder puzzles on average, especially for users with higher Elo. An interesting trend is that the recommendations typically underestimate the user Elo, which reflects the behavior of the behavior policy as well.

In Table 1, we summarize the results of the one-step importance sampling approaches compared to the behavior policy for different Elo buckets and growth groups. Across lower Elo buckets (between

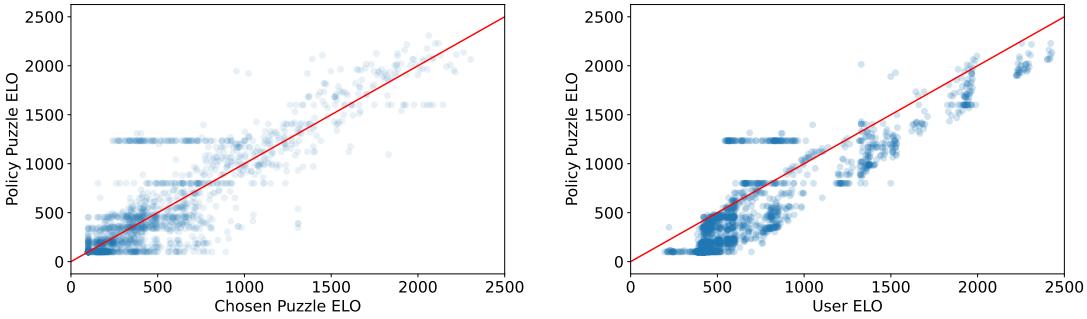


Figure 5: Distribution of trained policy's top recommended puzzle versus the puzzle Elo of the chosen puzzle (left) and the user Elo (right).

Elo	**Stagnant Group**		**Growth Group**		**Average**	
	π_b	π_θ	π_b	π_θ	π_b	π_θ
100–600	14.3 ± 1.5	$\mathbf{52.9 \pm 3.0}$	17.7 ± 0.4	$\mathbf{58.8 \pm 1.7}$	16.9 ± 0.5	$\mathbf{57.4 \pm 1.5}$
600–1000	14.1 ± 1.2	$\mathbf{27.0 \pm 6.7}$	20.9 ± 0.5	$\mathbf{42.3 \pm 3.2}$	19.5 ± 0.4	$\mathbf{39.2 \pm 2.9}$
1000–1500	12.3 ± 0.4	12.1 ± 1.2	18.3 ± 0.6	19.7 ± 2.4	15.3 ± 0.4	16.0 ± 1.4
1500+	12.2 ± 0.2	12.2 ± 1.2	15.9 ± 0.3	16.1 ± 1.1	13.5 ± 0.2	13.6 ± 0.9

Table 1: Returns obtained by trained policy π_θ and the Chess.com policy π_b through one-step importance sampling. We show 95% confidence intervals with $\pm$ and boldfaced numbers are statistically significant with Student's t-test.

100 and 1000), we show statistically significant and consistent improvement compared to the behavior policy. However, notably, as the Elo increases, our margins of improvement decrease, where our policy is neutral with respect to the behavior policy.

Qualitatively, this is sensible because optimal puzzle selection is more critical to learning at earlier stages, whereas missteps in puzzle recommendation may not lead to large outcome differences for an experienced chess player. Additionally, an important note is that the vast majority of the user base (and reflected in the training and evaluation data) is concentrated at lower Elo levels, with over 80% of users under 1500 Elo, where our improvement is roughly 110.1% over the existing system.

While the return is consistently lower for the stagnant group (as is expected), the relative performance improvement from the trained policy is approximately similar across growth groups and stagnant groups. That said, for the smallest Elo bucket, π_θ performs better relatively on stagnant group users, whereas for all others, π_θ performs relatively better on growth group users.

6.2 Expert Evaluation

We provide a qualitative analysis using the expert-designed rubrics and with the help of an LLM to understand whether there is any noticeable difference between the type of puzzles recommended by our learned policy and the original Chess.com puzzle serving system. We randomly sample 200 users from the holdout evaluation dataset. To evaluate the original Chess.com policy π_b, we directly use what was actually recommended to the user at that time. For the trained model π_θ, we sample the top-2 puzzles that have the highest probabilities of being recommended.

Our trained model recommends puzzles rated as slightly more enjoyable and slightly harder for the target player. The puzzles also receive somewhat higher calculation and pattern-recognition scores. Because these evaluations are generated by LLM judges calibrated on a small set of expert annotations, we treat the findings as preliminary.

Method	Calculation	Pattern Recognition	Fun	Rating	Quality	Informativeness
π_b (Chess.com)	70.17	58.89	65.57	67.95	69.83	75.11
π_θ (Ours)	72.39	61.48	70.34	72.73	71.31	75.11
Δ	+2.22	+2.59	+4.77**	+4.78*	+1.48	0.00

Table 2: Comparison of methods for qualitative expert-designed metrics. Metrics are converted to 100 from the original range to account for the difference in scale. Values are averaged across all annotated puzzles. We perform Student's t-test and use * to mark $p < 0.05$, and ** for $p < 0.01$.

7 Conclusion and Future Work

Our study leverages 1.5 billion puzzle-solving histories to learn the pedagogical value of chess puzzles and develop an automated puzzle selection system using offline reinforcement learning. Our offline policy evaluation and LLM-based qualitative analysis both suggest differences between the learned policy and Chess.com's existing policy. However, direct human evaluations are still needed to fully validate our system. In the future, we hope to collaborate with our data providers to demonstrate the actual impact on chess learners by using our system to filter out low-quality chess puzzles and to investigate whether our approach has applications in other domains, such as math, language learning, or coding.

Acknowledgment

During the period when this research was conducted, AN was supported in part by a Stanford HAI Hoffman-Yee grant and an NSF #2112926 grant. NT was funded by a grant from FAR.AI.

Paul Terwilliger was our main collaborator from Chess.com, where he served as the director of AI at the time. Paul provided us with the dataset and participated in multiple research meetings to discuss potential uses of the data. We are extremely grateful for his contribution to the project.

We also thank the rest of Chess.com for forming this collaboration with us and allowing us to explore different ways to empower chess learners all over the world.

Carissa Yip and Nicholas Tomlin served as our in-house chess experts. CY and NT co-designed the annotation schema used to rate each chess puzzle through an iterative process. CY helped recruit the expert chess annotators listed in Table 3. We are grateful to the annotators for volunteering approximately 1.5 hours to complete a set of annotations on our chess puzzles. Each annotator was compensated with a Coupa Cafe gift card or an Amazon gift card.

Name	**USCF Profile**	**Title**
Alexander Su	12857329	Expert
Robbie Selwyn	17281090	N/A
Iris Zhou	14467261	Expert
Tony Kukavica	13853062	National Master
Justin Paul	14323420	FIDE Master
Matt Larson	14278511	International Master
Bryce Tiglon	14230627	Grandmaster
Josiah Stearman	14006506	Grandmaster

Table 3: Volunteer chess annotators who completed puzzle annotations, with their USCF profile (linked via their USCF member ID) and title.

References

Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. Reinforcement learning for the adaptive

scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2020.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 456–464, 2019.

Chess.com. How we built a puzzle database with half a million puzzles, January 2023. URL https://www.chess.com/blog/CHESScom/how-we-built-a-puzzle-database-with-half-a-million-puzzles.

Min Chi, Pamela Jordan, Kurt Vanlehn, and Diane Litman. To elicit or to tell: Does it matter? In *Artificial Intelligence in Education*, pp. 197–204. IOS Press, 2009.

Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pp. 341–344, 2020.

K Anders Ericsson. Deliberate practice and acquisition of expert performance: a general overview. *Academic emergency medicine*, 15(11):988–994, 2008.

K Anders Ericsson, Ralf T Krampe, and Clemens Tesch-Römer. The role of deliberate practice in the acquisition of expert performance. *Psychological review*, 100(3):363, 1993.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. Chessgpt: Bridging policy learning and language modeling. *Advances in Neural Information Processing Systems*, 36:7216–7262, 2023.

Karim Hamade, Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Designing skill-compatible ai: Methodologies and frameworks in chess. *arXiv preprint arXiv:2405.05066*, 2024.

Xingwei He, Zhenghao Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.

Victor Henkin. *1000 Checkmate Combinations*. Batsford, 2021. ISBN 1849947252. URL https://www.amazon.com/1000-Checkmate-Combinations-Victor-Henkin/dp/1849947252.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, pp. 1645–1654. PMLR, 2017.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

Kenneth R Koedinger, Paulo F Carvalho, Ran Liu, and Elizabeth A McLaughlin. An astonishing regularity in student learning rate. *Proceedings of the National Academy of Sciences*, 120(13): e2221311120, 2023.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Aviral Kumar, Anikait Singh, Stephen Tian, Chelsea Finn, and Sergey Levine. A workflow for offline model-free robotic reinforcement learning. *arXiv preprint arXiv:2109.10813*, 2021.

Aviral Kumar, Rishabh Agarwal, Xinyang Geng, George Tucker, and Sergey Levine. Offline q-learning on diverse multi-task data both scales and generalizes. *arXiv preprint arXiv:2211.15144*, 2022.

Evan Liu, Moritz Stephan, Allen Nie, Chris Piech, Emma Brunskill, and Chelsea Finn. Giving feedback on interactive student programs with meta-exploration. *Advances in Neural Information Processing Systems*, 35:36282–36294, 2022.

Cong Lu, Philip J Ball, Tim GJ Rudner, Jack Parker-Holder, Michael A Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *arXiv preprint arXiv:2206.04779*, 2022.

Zhiyao Luo, Yangchen Pan, Peter Watkinson, and Tingting Zhu. Position: reinforcement learning in dynamic treatment regimes needs critical reexamination. *Journal of Machine Learning Research*, 2024.

Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, volume 1077, 2014.

Thomas McGrath, Andrei Kapishnikov, Nenad Tomašev, Adam Pearce, Martin Wattenberg, Demis Hassabis, Been Kim, Ulrich Paquet, and Vladimir Kramnik. Acquisition of chess knowledge in alphazero. *Proceedings of the National Academy of Sciences*, 119(47):e2206625119, 2022.

Reid McIlroy-Young, Yu Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Detecting individual decision-making style: Exploring behavioral stylometry in chess. *Advances in Neural Information Processing Systems*, 34:24482–24497, 2021.

Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning models of individual behavior in chess. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1253–1263, 2022.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Allen Nie, Ann-Katrin Reuel, and Emma Brunskill. Understanding the impact of reinforcement learning personalization on subgroups of students in math tutoring. In *International conference on artificial intelligence in education*, pp. 688–694. Springer, 2023.

Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*, 2019.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Henry L Roediger III, Adam L Putnam, and Megan A Smith. Ten benefits of testing and their applications to educational practice. *Psychology of learning and motivation*, 55:1–36, 2011.

Sherry Ruan, Allen Nie, William Steenbergen, Jiayu He, JQ Zhang, Meng Guo, Yao Liu, Kyle Dang Nguyen, Catherine Y Wang, Rui Ying, James Landay, and Emma Brunskill. Reinforcement learning tutor better supported lower performers in a math task. *Machine Learning*, pp. 1–26, 2024.

Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero. *arXiv preprint arXiv:2310.16410*, 2023.

Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. Saint+: Integrating temporal features for ednet correctness prediction. *arXiv preprint arXiv:2010.12042*, 2020.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022.

Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H Chi, and Minmin Chen. Surrogate for long-term user experience in recommender systems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 4100–4109, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

A Appendix

A.1 Human Annotation Guidelines

This appendix describes the annotation protocol used to collect the expert ratings discussed in Section 5. The goal of the annotation task is to measure whether puzzles recommended by our learned policy are of comparable or higher pedagogical quality than those served by Chess.com's existing tactics trainer. Each of the eight titled and expert-level annotators listed in Table 3 was given a spreadsheet of puzzle URLs, together with the Chess.com Elo of the player each puzzle was originally recommended to and that player's puzzle rating. Annotators were asked to first attempt to solve each puzzle and record whether they solved it correctly, before rating it along the six criteria below and providing a short written description and justification (minimum length one sentence) for their ratings. Annotators were asked to budget approximately 1.5 hours to complete the full set of 30 puzzles, inclusive of reading these guidelines, and were compensated with a gift card.

Calculation For each puzzle, annotators first judged whether it was primarily designed to test calculation, pattern recognition, both, or neither, and rated only the corresponding subcriterion below (marking the other as not applicable). *Calculation* is scored on a 1–4 scale and reflects the extent to which a puzzle requires reasoning over long move sequences or lines with a high branching factor; puzzles that can be solved by identifying the single forcing move at each step, without look-ahead, receive a low score. Figure 6a shows an example rated a 1, in which every move in the solution is the only reasonable continuation available at that step.

Pattern recognition *Pattern recognition* is scored on a 1–3 scale and reflects how unusual or interesting a puzzle's tactical motif is, relative to common motifs (e.g., a standard smothered mate); a puzzle earns the highest score either for testing a genuinely unusual pattern or for giving a common pattern an unusual twist.

Informativeness Scored on a 1–5 scale, *informativeness* measures whether a puzzle's given solution follows the most challenging and instructive line rather than a secondary, less demanding continuation. Figure 6b shows an example rated a 1, where the puzzle's solution avoids the most challenging continuation (which the solver must nonetheless calculate to justify an earlier move) in favor of an easier follow-up. Puzzles whose solutions instead follow the opponent's best defense, and whose logic remains clear once the solution is revealed, are rated highly.

(a) A *calculation* score of 1: for instance, in the below position, Black is forced to take on g6 and play Ng7, which solves the puzzle.

(b) An *informativeness* score of 1: for instance, in the below position, after we take on h3, the solution follows 2. Qg2 instead of taking on f5 (the most challenging line, which the player must have calculated before taking on h3).

Figure 6: Example puzzle positions used to illustrate the *calculation* (Figure 6a) and *informativeness* (Figure 6b) criteria.

Rating-appropriateness Scored on a 1–5 scale, *rating-appropriateness* captures the annotator's assessment of a puzzle's difficulty relative to the rating of the player it was recommended to. Annotators were asked to judge this criterion from the perspective of a player at the target rating, rather than their own.

Fun Also scored on a 1–5 scale, *fun* measures how enjoyable a puzzle would likely be for a player at the target rating.

Quality *Quality* is a holistic 1–5 score of the puzzle as a whole, independent of difficulty; the preceding criteria are intended to inform this judgment without strictly determining it. Figure 4 summarizes the rating range for each criterion.

A.2 LLM Judge Prompt Fields

Each LLM judge receives a system message and a DSPy signature. The system message identifies the expert annotator being imitated by their chess title, when applicable, and USCF Elo rating, and asks the model to apply the annotation guidelines together with that expert-level perspective. The signature contains the following three input fields:

- *Annotation guidelines*: the complete rubric for rating calculation, pattern recognition, informativeness, rating-appropriateness, quality, and fun.
- *Target-user puzzle Elo*: the target user's current Chess.com puzzle Elo, which provides the reference skill level for rating-appropriateness and fun.
- *Puzzle board*: an ASCII grid representation of the position, generated from the puzzle's FEN string, followed by a sentence describing the puzzle's first move and asking the solver to find the best continuation.

The signature's output field requests integer scores from 1 to 100 for all six categories in valid JSON format. DSPy also supplies selected expert-rated examples when executing the optimized judge.

A.3 Annotation Questionnaire

Annotators recorded their ratings in a shared spreadsheet, with one puzzle per row and one column per question. For each puzzle, the columns asked, verbatim:

- *What is the primary emphasis of this puzzle?* (Calculation / Pattern Recognition / Both / Neither)
- *Calculation*: this puzzle requires the player to think through the entire solution before playing their first move. (1–4, or N/A)
- *Pattern recognition*: this puzzle teaches useful or un-ordinary motifs, e.g. the motifs are not banal/overdone, or if they are, the puzzle puts a unique spin on them. (1–3, or N/A)
- *Informativeness*: the solution goes down "the right path" to make sure the player understands the point of the puzzle. (1–5)
- *Rating*: is this puzzle appropriate for someone rated at the target player's skill level? (1–5)
- *Quality*: this is a high-quality puzzle, regardless of whether it is at the correct Elo level or not. (1–5)
- *Fun*: I think this puzzle would be enjoyable and fun to solve by players at the target Elo. (1–5)
- *Additional notes*: please write a 1–2 sentence justification of your ratings.

Table 4 reproduces two rows from a completed annotation spreadsheet to illustrate how one annotator answered this questionnaire in practice.

A.4 Selecting Illustrative Examples

To illustrate how ratings can differ across puzzles, and to qualitatively showcase the difference between puzzles recommended by our system and the original system, we selected a second pair of example puzzles, shown in Table 5. As with Table 4, we restricted our search to the subset of puzzles that all eight annotators independently rated (the "preference learning" puzzles described in Section 5). Within this subset, we computed each puzzle's average pattern recognition score (excluding annotators for whom the criterion did not apply) and selected one puzzle near the top and one near the bottom of this ranking, from our policy and from Chess.com's existing system respectively.

The first example (a puzzle recommended by our policy) received the maximum pattern recognition score from every annotator who rated it: all described the position's combination of moves as an unusual or non-obvious way to reach the tactical idea, distinct from a puzzle that merely tests whether the solver knows a standard pattern.

Field	**Example 1**	**Example 2**
Puzzle	#1004456	#955974
Player Elo (Puzzle Elo)	951 (1317)	1361 (1615)
Solved correctly	Yes	Yes
Primary emphasis	Pattern Recognition	Both
Calculation	N/A	1 – easy to guess first move w/o seeing solution
Pattern recognition	1 – banal/boring pattern	1 – banal/boring pattern
Informativeness	5 – most informative	1 – least informative
Rating-appropriateness	4 – a bit too hard	3 – just right
Quality	3	1 – lowest quality
Fun	4	2
Annotator notes	*"Rated as 4 difficulty because it is winning a piece not mate, and seeing the fork idea/long range check may be challenging for the rating."*	*"The puzzle if done properly is about right in terms of difficulty. As is, it is bad. The puzzle does not even give a line, which is horrifically bad, and makes it easy to guess the correct answer without full understanding/calculation."*

Table 4: Two example rows from a completed annotation spreadsheet, showing how one annotator answered the questionnaire in Appendix A.3 for two puzzles.

The second example (a puzzle served by Chess.com's existing system) had a lower average pattern recognition score, with annotators ranging from "banal/boring pattern" to "interesting and useful" for the same puzzle. Notably, its holistic quality score remained comparable to the first example's, since most annotators still considered it a reasonably solid puzzle overall; only the pattern recognition criterion, and the accompanying free-text notes, surfaced the disagreement over whether its underlying motif was novel or overdone. This is precisely the gap we selected these examples to illustrate: a puzzle can score well on quality while still being seen by some experts as testing an unremarkable, overdone pattern.

Table 6 shows the full pattern recognition and quality distributions across all eight annotators for both puzzles, rather than the mean alone, so that the degree of agreement (or disagreement) on each criterion is directly visible.

Field	Example 1 (Ours)	Example 2 (Chess.com)
Puzzle	#760014	#1072600
Player Elo (Puzzle Elo)	2175 (3053)	951 (1317)
Solved correctly	Yes	Yes
Primary emphasis	Both	Calculation
Calculation	4 – requires full understanding of solution to solve	2
Pattern recognition	3 – interesting and useful	1 – banal/boring pattern
Informativeness	5 – most informative	4
Rating-appropriateness	3 – just right	2 – a bit too easy
Quality	5 – highest quality	3
Fun	5 – most fun	2
Annotator notes	*"A+ for fun and useful. Also, b5+ and Rd5+ is a particularly unusual combination which requires calculation... as well as nice checking pattern recognition. Good puzzle!"*	*"A bit of a boring problem which forces White to find all checks. It is possible that White forgets to give the checks and creates the mating threat with 1.Qf7, which at this level where checks are the first thing to examine the problem isn't too difficult."*

Table 5: Two example rows from a completed annotation spreadsheet, selected to contrast a puzzle with a unanimously high pattern recognition score (Example 1) against a puzzle with a lower, more disputed pattern recognition score (Example 2), as discussed in Appendix A.4.

Annotator	Pattern Recognition		Quality	
	Ours	Chess.com	Ours	Chess.com
A	3	2	3	4
B	N/A	2	5	4
C	N/A	N/A	4	5
D	3	1	5	3
E	3	3	3	4
F	3	3	5	5
G	N/A	N/A	5	4
H	3	3	4	4
Mean	3.00	2.33	4.25	4.12

Table 6: Pattern recognition and quality scores given by each of the eight annotators to the two puzzles in Table 5 ("Ours" is #760014 and "Chess.com" is #1072600; N/A denotes an annotator for whom the pattern recognition criterion did not apply), with rows aligned so that the same letter denotes the same annotator across all four columns. While the two puzzles' quality scores are similar on average, their pattern recognition scores diverge, with #1072600 receiving both the single lowest score (a "banal/boring pattern" rating) and a wider spread of opinions overall.